Learning goals

PROOF OF FALSE

THEOREM: For any positive integer \( n \), there exists a justification for going \( n \) days without showering.

PROOF: We proceed by induction on \( n \).

BASE CASE: \( n = 1 \)
Most people shower once per day. Right before you shower, you have gone 1 day without showering. If going 1 day without showering were unreasonable, you would have showered before then.

INDUCTION STEP
We assume the induction hypothesis is true for \( n \) and now consider the case \( n + 1 \). Suppose you have a problem set or an exam the next day. Clearly, showering can wait 1 more day until you get that done.

QED
I'm not going near you ever again.

by Andrew Spann

THE BEST PART IS I USE THE SAME ARGUMENT TO JUSTIFY NOT GOING TO SLEEP FOR JUST 1 MORE HOUR.
About you

Have you used iClickers before?
A. Yes
B. No

To change your remote frequency
1. Press and hold power button until flashing
2. Enter two-letter code
3. Checkmark / green light indicates success

CENTR109: AC       WLH2001: CA

Why use clickers?
About you

What other CSE class are you taking this quarter?

A. None.
B. CSE 12.
C. CSE 11.
D. CSE 8B.
E. Some other CSE class.

To change your remote frequency
1. Press and hold power button until flashing
2. Enter two-letter code
3. Checkmark / green light indicates success

CENTR109: AC
WLH2001: CA
Introductions
Logistics

Textbook: Rosen 7th Edition

Participation: Class times (iClicker questions) & discussion (quizzes)
https://www1.iclicker.com/register-clicker/
http://sections.ucsd.edu

Exams:
Tuesday January 26
Tuesday February 23
** No makeup exams **
Exam week – check schedule of classes

Gradescope: Homework submission and exam return
Piazza: announcements and Q&A. Contact instructors here!
Office hours: instructors and tutors. Discuss HW questions here!
Class podcast: podcast.ucsd.edu
Logistics

• Exams 65%  HW + Participation 35%
• Details on class website: [http://cseweb.ucsd.edu/classes/wi16/cse20-ab/](http://cseweb.ucsd.edu/classes/wi16/cse20-ab/)
• Weekly HW, can be done in groups of 1-3, can change groups throughout quarter
• Drop lowest HW score
• Drop lowest midterm score if do better on final
• Can use note sheet for exams
• Participation earned via either class participation or discussion quizzes
• Drop lowest discussion quiz score and two lowest class participation scores
• Credit for participation if answer 80% of clicker question in that day's class
• HW and exams answers evaluated not only on the correctness of your answers, but on your ability to effectively communicate your ideas and convince the reader of your conclusions through proofs and logical reasoning
• Assume some familiarity with programming (not necessarily specific language)
You are working on a homework question with your group members and are stuck on a question. You run into a friend who solved the problem already and shows you her solution. You look at it, but put it away before continuing the group conversation. Is this acceptable?

A. Yes
B. No
About this class: Academic integrity

You form a group to work on a homework assignment. There are three group members and three questions. You split up the work so that each student gets one question. After you work on your individual questions for a while, you get together as a group and proofread the solutions, then hand them in. Is this acceptable?

A. Yes
B. No
About this class: Academic integrity

You’re working on a homework question and run across a definition you don’t understand. You Google the term and, ‘lo and behold, the first hit is a full solution to the homework question. You avoid reading the solution and close the browser. You keep working on the solution and hand in the assignment, without mentioning the Google search since you didn’t use the result. Is this acceptable?

A. Yes
B. No
About this class: Academic integrity

You're not sure if you are interpreting a homework problem correctly. You write a post on Piazza explaining your approach to answering it, and asking if this is the correct way to interpret the question. Is this acceptable?

A. Yes
B. No
Today's learning goals

• Trace pseudocode given input.
• Explain the higher-level function of an algorithm expressed with pseudocode.
• Identify and explain (informally) whether and why given pseudocode satisfies properties of being an algorithm.
• Define the greedy approach for an optimization problem.
• Analyze whether the greedy approach solves an optimization problem.
Algorithms!

Multiply 142 x 17.
What did we do?
Another way? RPM

Write the factors in two columns.

Repeatedly double the LEFT and halve the right. (Truncate fractions, i.e. toss remainders)

Cross out the LEFT values where the RIGHT values are even.

Add the remaining LEFT values together.
Is there another way? RPM

Write the factors in two columns.

Repeatedly double the LEFT and halve the right. (Truncate fractions, i.e. toss remainders)

Cross out the LEFT values where the RIGHT values are even.

Add the remaining LEFT values together.
Algorithm?

Finite sequence of precise instructions for solving problem.

Rosen 3.1 p. 191
Algorithm?

Finite sequence of precise instructions for solving problem.

Finiteness

Definiteness

Correctness

Rosen 3.1 p. 191
To describe an algorithm

- English description.

- Pseudocode.
To describe an algorithm

- English description.

- Pseudocode.

For RPM: see HW1
To describe an algorithm

procedure alg1(a : real number; n : positive integer)

1. \( x := 1.0 \)
2. for \( i := 1 \) to \( n \)
3. \( x := x \cdot a \)
4. return \( x \)

What is / are the input(s)?

A. Only \( x \)
B. Only \( a \) and \( n \)
C. Only \( a \) and \( x \)
D. Only \( x \) and \( n \)
E. The variables \( a, i, x \) and \( n \)
To describe an algorithm

\textbf{procedure} \textit{alg1}(a \ : \ real \ number; \ n \ : \ positive \ integer)\\

1. \text{ } x := 1.0\\
2. \textbf{for } i := 1 \textbf{ to } n\\
3. \text{ } x := x \cdot a\\
4. \textbf{return} \ x

\textit{Definite? Finite?}
To describe an algorithm

procedure $alg1(a: \text{ real number}; n: \text{ positive integer})$

1. $x := 1.0$ Assignment
2. for $i := 1$ to $n$
3. $x := x \cdot a$
4. return $x$

Definite? Finite?
To describe an algorithm

procedure \textit{alg1}(a: \text{real number}; n: \text{positive integer})

1. \(x := 1.0\)
2. \textbf{for} \(i := 1\ \textbf{to} \ n\)
3. \(x := x \cdot a\)
4. \textbf{return } x

What does the procedure return when \(a\) is 3 and \(n\) is 4?

A. 1  
B. 12  
C. 64  
D. 81  
E. None of the above.
To describe an algorithm

procedure \textit{alg1}(a : real number; n : positive integer)

1. \( x := 1.0 \)
2. \textbf{for} \( i := 1 \) \textbf{to} \( n \)
3. \( x := x \cdot a \)
4. \textbf{return} \( x \)

What's a description in English of this algorithm?
To describe an algorithm

\textbf{procedure} \textit{alg1}(a : real number; n : positive integer)

1. \( x := 1.0 \)
2. \textbf{for} \( i := 1 \) \textbf{to} \( n \)
3. \( x := x \cdot a \)
4. \textbf{return} \( x \)

Describe the output of this algorithm as a function of \( a \) and \( n \).
To describe an algorithm

procedure $alg2(a : \text{ real number}; n : \text{ positive integer})$

1. $x := 1.0$
2. $i := n$
3. while $i > 0$
4. if $i \mod 2 = 1$ then $x := x \cdot a$
5. $i := i \div 2$
6. if $i > 0$ then $a := a \cdot a$
7. return $x$
To describe an algorithm

procedure alg2(a : real number; n : positive integer)

1. $x := 1.0$
2. $i := n$
3. while $i > 0$
4. if $i \mod 2 = 1$ then $x := x \cdot a$
5. $i := i \div 2$
6. if $i > 0$ then $a := a \cdot a$
7. return $x$

What does the procedure return when $a$ is 3 and $n$ is 4?

A. 1
B. 12
C. 64
D. 81
E. None of the above.
To describe an algorithm

procedure \textit{alg}2(a : real number; n : positive integer)

1. \(x := 1.0\)
2. \(i := n\)
3. \textbf{while} \(i > 0\)
4. \textbf{if} \(i \mod 2 = 1\) \textbf{then} \(x := x \cdot a\)
5. \(i := i \div 2\)
6. \textbf{if} \(i > 0\) \textbf{then} \(a := a \cdot a\)
7. \textbf{return} \(x\)
To describe an algorithm

procedure *alg2*(*a*: real number; *n*: positive integer)

1.  \(x := 1.0\)
2.  \(i := n\)
3.  **while** \(i > 0\)
4.  \[\text{if } i \text{ mod } 2 = 1 \text{ then } x := x \cdot a\]
5.  \(i := i \div 2\)
6.  \[\text{if } i > 0 \text{ then } a := a \cdot a\]
7.  **return** \(x\)
Algorithm?

Finite sequence of precise instructions for solving problem.

Not just arithmetic!

Rosen 3.1 p. 191
Optimization
Cookies!

Select 6 cookies

Optimize: maximize total number of chocolate chips
Cookies!

Select 6 cookies

Optimize: maximize total number of chocolate chips

**Greedy approach:** pick cookie with most chocolate chips first, then with 2\textsuperscript{nd} most second, etc.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td>76</td>
<td>69</td>
<td>60</td>
<td>75</td>
<td>51</td>
</tr>
<tr>
<td>61</td>
<td>77</td>
<td>74</td>
<td>72</td>
<td>80</td>
<td>58</td>
</tr>
<tr>
<td>82</td>
<td>97</td>
<td>94</td>
<td>88</td>
<td>99</td>
<td>92</td>
</tr>
<tr>
<td>47</td>
<td>68</td>
<td>59</td>
<td>52</td>
<td>65</td>
<td>40</td>
</tr>
<tr>
<td>78</td>
<td>81</td>
<td>79</td>
<td>71</td>
<td>85</td>
<td>62</td>
</tr>
<tr>
<td>50</td>
<td>67</td>
<td>73</td>
<td>57</td>
<td>70</td>
<td>46</td>
</tr>
</tbody>
</table>
Cookies!

Select 6 cookies

Optimize: maximize total number of chocolate chips

Greedy approach: pick cookie with most chocolate chips first, then with 2\textsuperscript{nd} most second, etc.

Definite? Finite? Correct? i.e. Will it always find the optimal solution?
Cookies!

Select 6 cookies at most one from each row and column

Optimize: maximize total number of chocolate chips

Greedy approach: pick cookie with most chocolate chips first, then with 2nd most second, etc.
Cookies!

Which cookies will be selected by the greedy algorithm?

<table>
<thead>
<tr>
<th></th>
<th>A.</th>
<th></th>
<th>B.</th>
<th></th>
<th>C.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td>76</td>
<td>69</td>
<td>60</td>
<td>75</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>77</td>
<td>74</td>
<td>72</td>
<td>80</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>97</td>
<td>94</td>
<td>88</td>
<td>99</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>68</td>
<td>59</td>
<td>52</td>
<td>65</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>81</td>
<td>79</td>
<td>71</td>
<td>85</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>67</td>
<td>73</td>
<td>57</td>
<td>70</td>
<td>46</td>
<td></td>
</tr>
</tbody>
</table>

A. None of the above.
Cookies!

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td>76</td>
<td>69</td>
<td>60</td>
<td>75</td>
<td>51</td>
</tr>
<tr>
<td>61</td>
<td>77</td>
<td>74</td>
<td>72</td>
<td>80</td>
<td>58</td>
</tr>
<tr>
<td>82</td>
<td>97</td>
<td>94</td>
<td>88</td>
<td>99</td>
<td>92</td>
</tr>
<tr>
<td>47</td>
<td>68</td>
<td>59</td>
<td>52</td>
<td>65</td>
<td>40</td>
</tr>
<tr>
<td>78</td>
<td>81</td>
<td>79</td>
<td>71</td>
<td>85</td>
<td>62</td>
</tr>
<tr>
<td>50</td>
<td>67</td>
<td>73</td>
<td>57</td>
<td>70</td>
<td>46</td>
</tr>
</tbody>
</table>
```

Select 6 cookies at most one from each row and column

Optimize: maximize total number of chocolate chips

**Greedy approach:** pick cookie with most chocolate chips first, then with 2\textsuperscript{nd} most second, etc.

*May not find the optimal solution! Proof: counterexample.*
Cookies!

<table>
<thead>
<tr>
<th></th>
<th>56</th>
<th>76</th>
<th>69</th>
<th>60</th>
<th>75</th>
<th>51</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td>77</td>
<td>74</td>
<td>72</td>
<td>80</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>97</td>
<td>94</td>
<td>88</td>
<td>99</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>68</td>
<td>59</td>
<td>52</td>
<td>65</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>81</td>
<td>79</td>
<td>71</td>
<td>85</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>67</td>
<td>73</td>
<td>57</td>
<td>70</td>
<td>46</td>
<td></td>
</tr>
</tbody>
</table>

**Greedy choice:**
gets $99+81+74+60+50+40 = 404$

<table>
<thead>
<tr>
<th></th>
<th>56</th>
<th>76</th>
<th>69</th>
<th>60</th>
<th>75</th>
<th>51</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td>77</td>
<td>74</td>
<td>72</td>
<td>80</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>97</td>
<td>94</td>
<td>88</td>
<td>99</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>68</td>
<td>59</td>
<td>52</td>
<td>65</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>81</td>
<td>79</td>
<td>71</td>
<td>85</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>67</td>
<td>73</td>
<td>57</td>
<td>70</td>
<td>46</td>
<td></td>
</tr>
</tbody>
</table>

**Optimal choice:**
gets $92+78+75+73+72+68 = 458$

*JS p. 77, Rosen p. 198*
For next time

- Start Homework 1
  - Set up course tools
  - Pseudocode and algorithms